Introduction to Data Science

Assignment Part 2

GROUP 79

442575 FAHIMEH FEREYDOUNIAN

442589 AROMA AGARWAL

JOOST-HENNING GROOT BRAMEL

1 Frequent Itemsets and Association Rules

1a) The three items which generated the highest sales value:

Itemname	**SalesValuePerItem**
POSTAGE	59489.5
REGENCY CAKESTAND 3 TIER	18961.8
RABBIT NIGHT LIGHT	18836.0

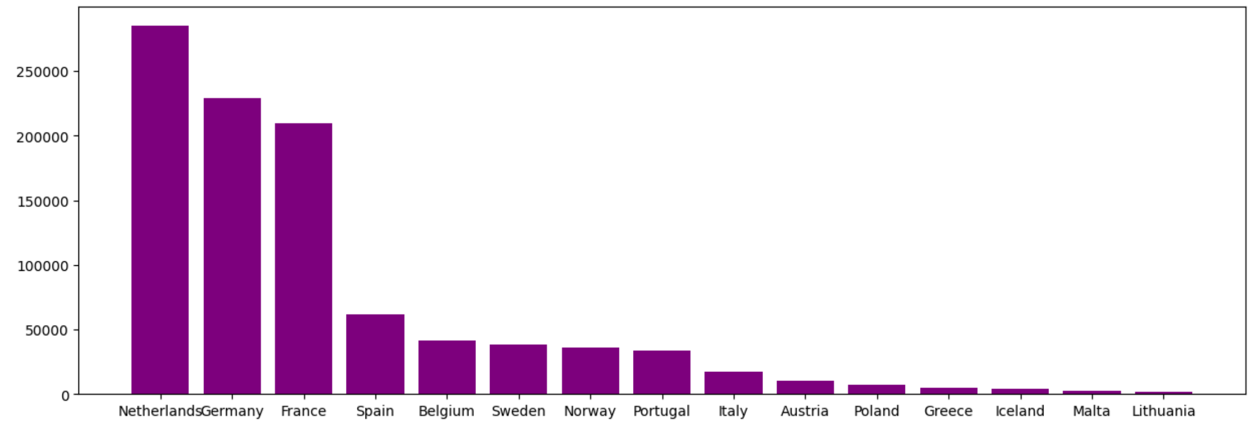


Fig. 1.1: Plot showing the sales value per country

1b) Number of transactions: 1357

Three most common items across all orders and their support:

Itemname	**Support**
POSTAGE	0.724392
ROUND SNACK BOXES SET OF4 WOODLAND	0.196021
REGENCY CAKESTAND 3 TIER	0.137067

Bar chart, which shows the support per item (except the most frequent one) over all transactions in descending order:

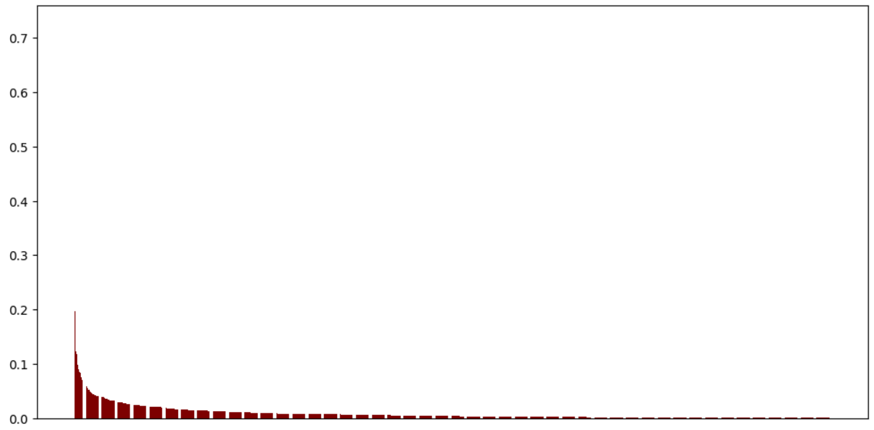


Fig. 1.2: Support per item (except the most frequent one)

1c) Yes, they are not the same, although they contain mutual items (which are not in the same rank). The most frequent items are counted by the number of transactions that contain them without considering the count of that item in each transaction. On the other hand, the highest sales values are calculated based on the item's price and quantity.

Usually the most frequent items are the items that have the lowest price and are sold in high quantities. So, the most frequent items are not necessarily the items that generated the highest sales values.

1d) The three most frequent itemsets which include at least three items and their support:

Itemsets	**Support**
Postage, Round Snack Boxes Of 4 Fruits, Round Snack Boxes Of4 Woodland	0.080324
Plasters In Tin Circus Parade, Postage, Plasters In Tin Woodland Animals	0.053795
Plasters In Tin Spaceboy, Postage, Plasters In Tin Woodland Animals	0.050847

1e) The three most frequent maximal itemsets which include at least three items and their support:

Itemsets	**Support**
Red Toadstool Led Night Light, Postage, Rabbit Night Light	0.022108

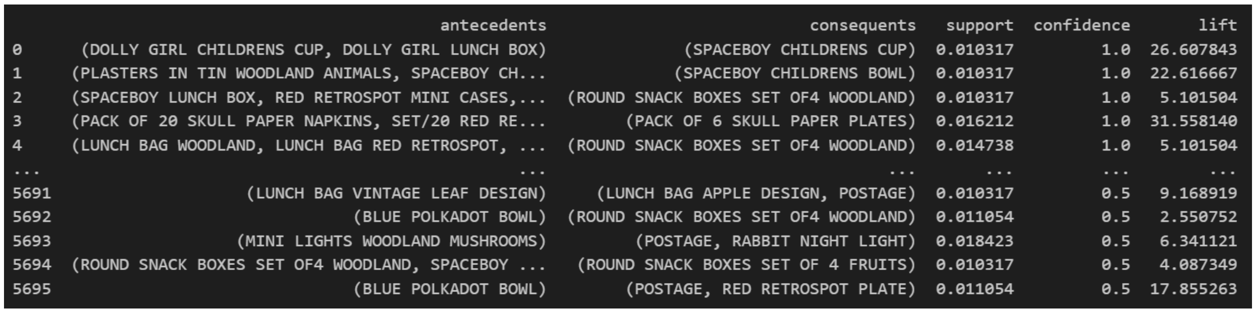
Blue Harmonica In Box, Red Harmonica In Box, Postage	0.020634
Postage, Rabbit Night Light, Round Snack Boxes Set Of4 Woodland	0.020634

The three most frequent maximal itemsets (from part (e)) have lower support than the frequent itemsets (from part (d)) because maximal itemsets are the largest itemsets that can't be extended while remaining frequent. Frequent itemsets include all combinations above the minimum support threshold, and adding items typically decreases support, which is why maximal itemsets tend to have lower support.

1f)

Length	**Frequent**	**Closed**	**Maximal**
6	8	5	1
7	1	1	1

1g)



```
                                          antecedents                            consequents   support  confidence       lift
0      (DOLLY GIRL CHILDRENS CUP, DOLLY GIRL LUNCH BOX)             (SPACEBOY CHILDRENS CUP)  0.010317         1.0  26.607843
1     (PLASTERS IN TIN WOODLAND ANIMALS, SPACEBOY CH...            (SPACEBOY CHILDRENS BOWL)  0.010317         1.0  22.616667
2     (SPACEBOY LUNCH BOX, RED RETROSPOT MINI CASES,...  (ROUND SNACK BOXES SET OF4 WOODLAND)  0.010317         1.0   5.101504
3     (PACK OF 20 SKULL PAPER NAPKINS, SET/20 RED RE...        (PACK OF 6 SKULL PAPER PLATES)  0.016212         1.0  31.558140
4     (LUNCH BAG WOODLAND, LUNCH BAG RED RETROSPOT, ...  (ROUND SNACK BOXES SET OF4 WOODLAND)  0.014738         1.0   5.101504
...                                                 ...                                  ...       ...         ...        ...
5691                      (LUNCH BAG VINTAGE LEAF DESIGN)     (LUNCH BAG APPLE DESIGN, POSTAGE)  0.010317         0.5   9.168919
5692                                (BLUE POLKADOT BOWL)  (ROUND SNACK BOXES SET OF4 WOODLAND)  0.011054         0.5   2.550752
5693                     (MINI LIGHTS WOODLAND MUSHROOMS)          (POSTAGE, RABBIT NIGHT LIGHT)  0.018423         0.5   6.341121
5694  (ROUND SNACK BOXES SET OF4 WOODLAND, SPACEBOY ...     (ROUND SNACK BOXES SET OF 4 FRUITS)  0.010317         0.5   4.087349
5695                                (BLUE POLKADOT BOWL)        (POSTAGE, RED RETROSPOT PLATE)  0.011054         0.5  17.855263
```

Fig. 1.3: All association rules with a minimum confidence of 0:5.

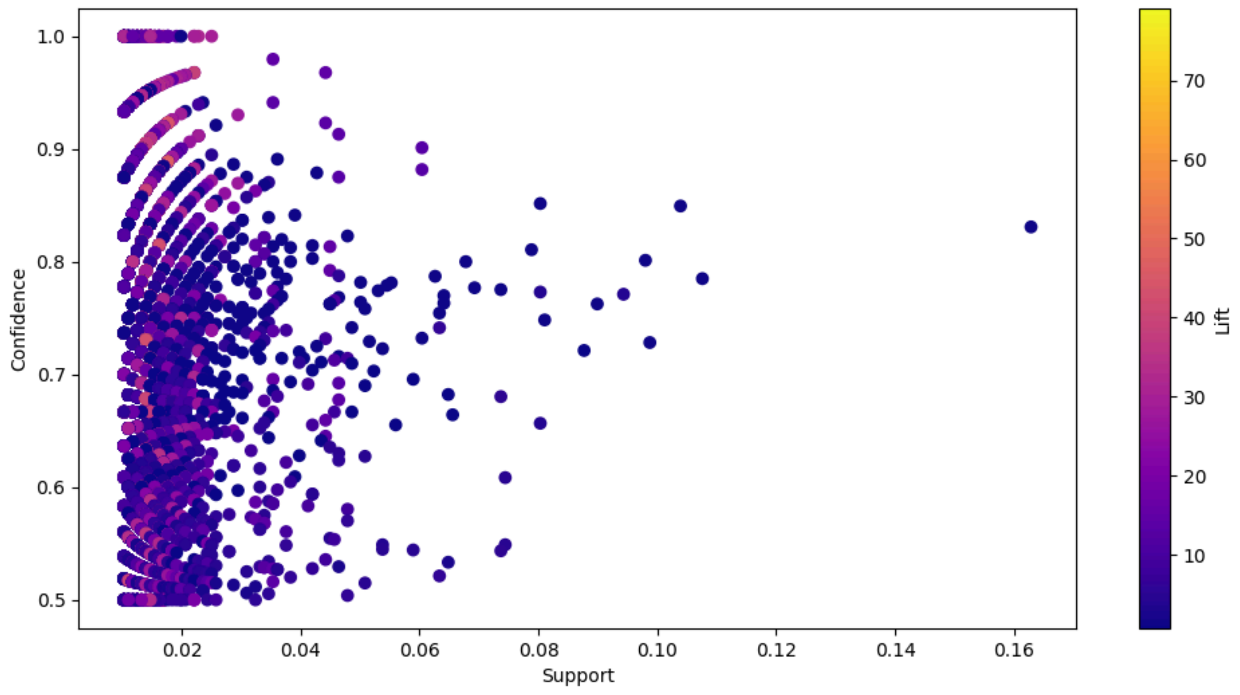
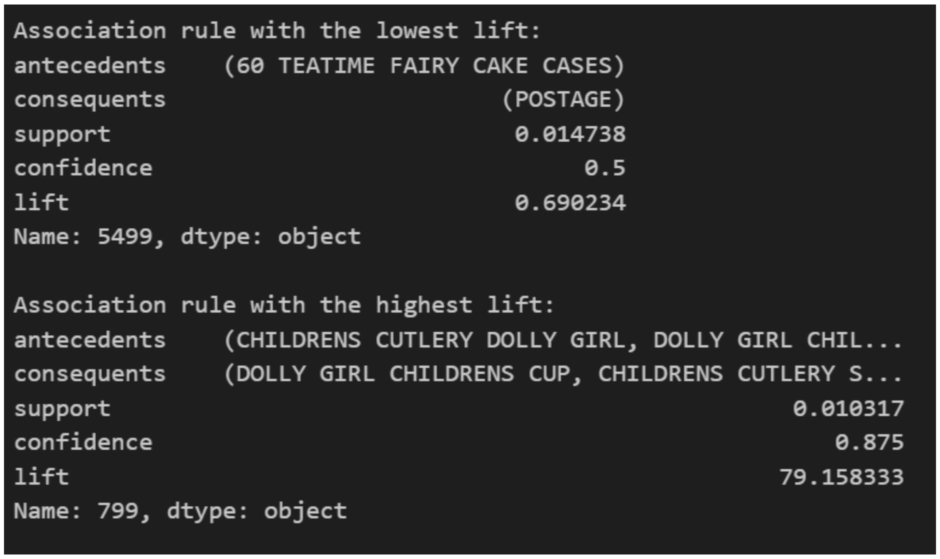


Fig. 1.4: Scatter plot of association rules with a minimum confidence of 0:5.

1h)



```
Association rule with the lowest lift:
antecedents    (60 TEATIME FAIRY CAKE CASES)
consequents                        (POSTAGE)
support                             0.014738
confidence                               0.5
lift                                0.690234
Name: 5499, dtype: object

Association rule with the highest lift:
antecedents    (CHILDRENS CUTLERY DOLLY GIRL, DOLLY GIRL CHIL...
consequents    (DOLLY GIRL CHILDRENS CUP, CHILDRENS CUTLERY S...
support                                                 0.010317
confidence                                                 0.875
lift                                                   79.158333
Name: 799, dtype: object
```

Fig. 1.5: Association rules with maximal lift values.

Association Rule with the Lowest Lift

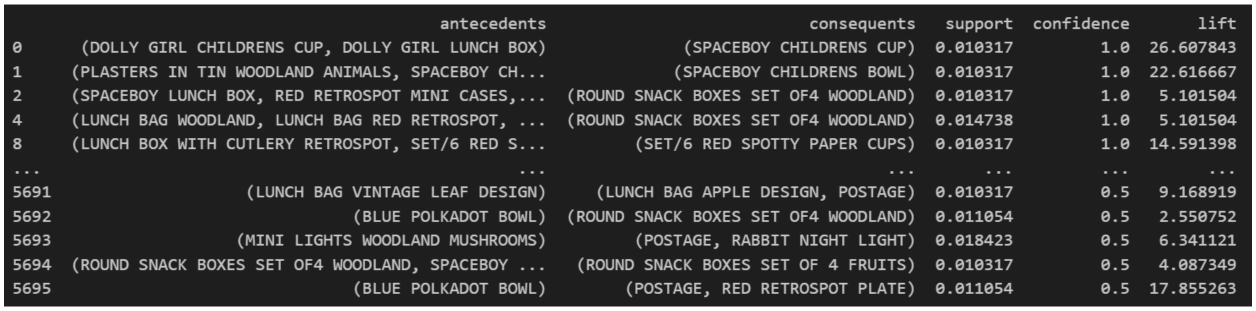
Support: 1.47% of transactions include both items. While the rule is valid, its limited occurrence in transactions makes it less significant overall.

Confidence: 50% of transactions with TEATIME FAIRY CAKE CASES also have POSTAGE. Confidence is moderate, but lift shows the relationship is likely coincidental.
Lift: 0.69 < 1, indicating a negative dependence. These items are less likely to appear together than expected if independent.

Association Rule with the Highest Lift
Support: 1.03% of transactions include both itemsets.
Confidence: 87.5% of transactions with antecedents also have the consequents. This suggests a very strong and reliable relationship, indicating that customers purchasing the antecedents are highly likely to purchase the consequents.
Lift: 79.16 >> 1, indicating a very strong positive dependence. This means that the items in the consequent are about 79 times more likely to be purchased with the antecedent than expected under independence. These items are highly likely to be purchased together, suggesting a strong association for bundling or targeted marketing.

1i)
Confidence: ≥ 0.7 to ensure the recommendation is frequent and occurs more often than not. It ensures that the rule is reliable in occurring.
Lift: ≥ 2 to ensure that the association rule between the antecedent and consequent is strong. Meaning they are bought together with a high probability and not due to random chance.

1j)



```
                                          antecedents                          consequents   support  confidence       lift
0      (DOLLY GIRL CHILDRENS CUP, DOLLY GIRL LUNCH BOX)            (SPACEBOY CHILDRENS CUP)  0.010317         1.0  26.607843
1     (PLASTERS IN TIN WOODLAND ANIMALS, SPACEBOY CH...           (SPACEBOY CHILDRENS BOWL)  0.010317         1.0  22.616667
2     (SPACEBOY LUNCH BOX, RED RETROSPOT MINI CASES,...  (ROUND SNACK BOXES SET OF4 WOODLAND)  0.010317         1.0   5.101504
4     (LUNCH BAG WOODLAND, LUNCH BAG RED RETROSPOT, ...  (ROUND SNACK BOXES SET OF4 WOODLAND)  0.014738         1.0   5.101504
8     (LUNCH BOX WITH CUTLERY RETROSPOT, SET/6 RED S...         (SET/6 RED SPOTTY PAPER CUPS)  0.010317         1.0  14.591398
...                                                 ...                                 ...       ...         ...        ...
5691                    (LUNCH BAG VINTAGE LEAF DESIGN)      (LUNCH BAG APPLE DESIGN, POSTAGE)  0.010317         0.5   9.168919
5692                                (BLUE POLKADOT BOWL)  (ROUND SNACK BOXES SET OF4 WOODLAND)  0.011054         0.5   2.550752
5693                   (MINI LIGHTS WOODLAND MUSHROOMS)         (POSTAGE, RABBIT NIGHT LIGHT)  0.018423         0.5   6.341121
5694  (ROUND SNACK BOXES SET OF4 WOODLAND, SPACEBOY ...   (ROUND SNACK BOXES SET OF 4 FRUITS)  0.010317         0.5   4.087349
5695                                (BLUE POLKADOT BOWL)        (POSTAGE, RED RETROSPOT PLATE)  0.011054         0.5  17.855263
```

Fig. 1.6: Rules such that antecedent and consequent form a maximal frequent itemset.

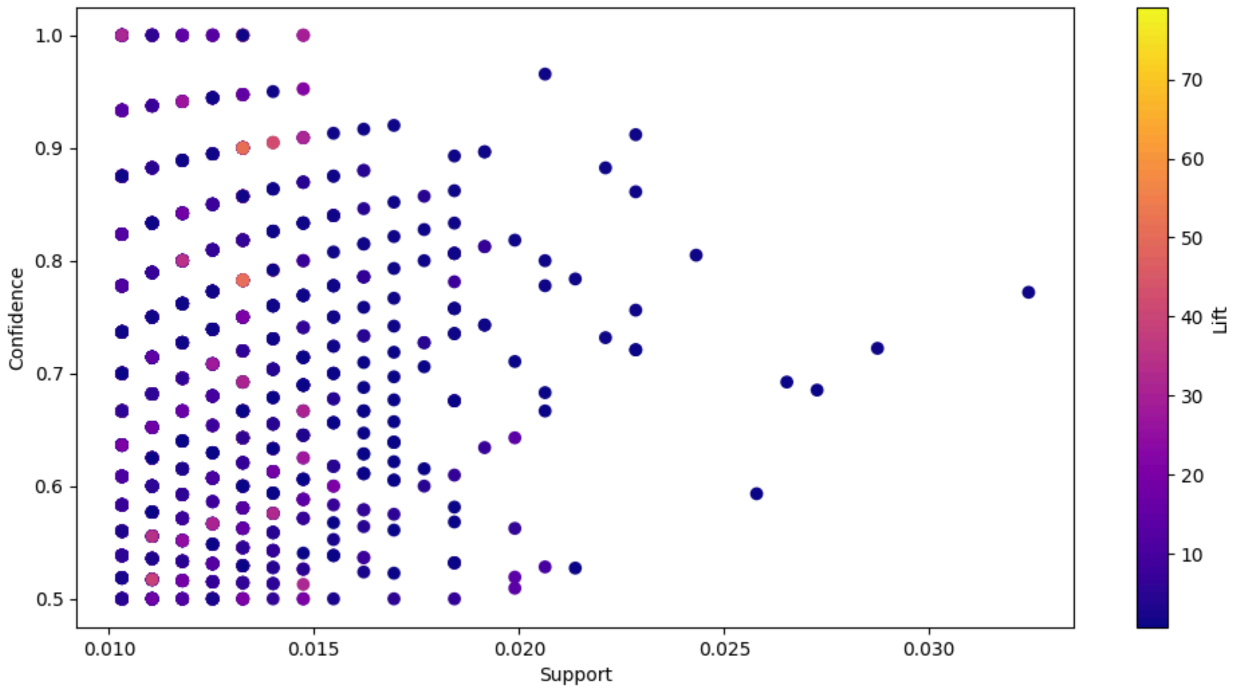


Fig. 1.7: Scatter plot of maximal frequent itemset

The maximum values have decreased after filtering, notably for support. This is because maximal frequent itemsets are the largest itemsets that can't be extended while remaining frequent, which typically results in lower support values.

The confidence and lift values also decrease because maximal itemsets tend to have lower support, which affects the confidence and lift values of the association rules. Here we notice lower maximum points in the scatter plot as well.

2: Text Mining

2a)

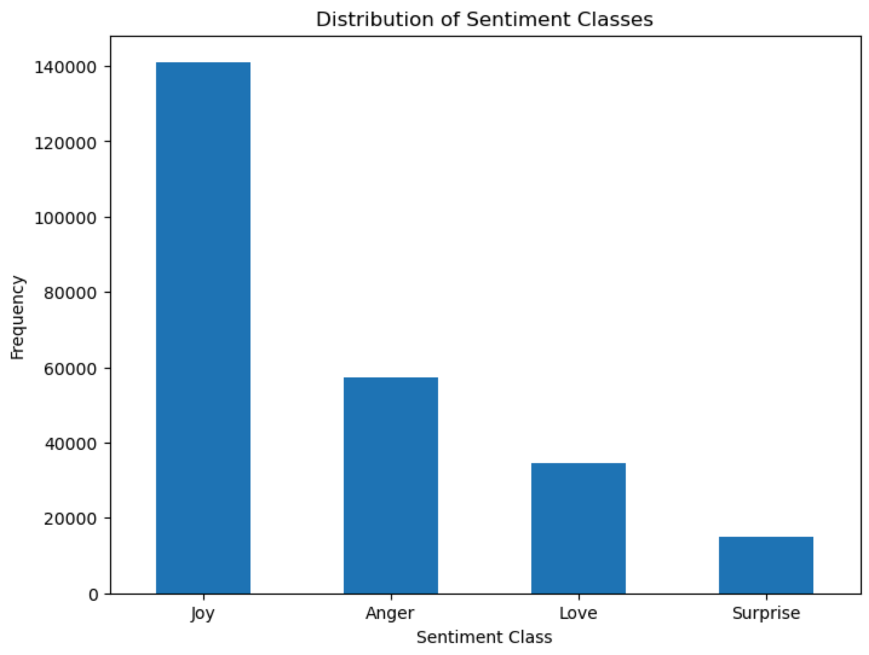


Fig. 2.1 Distribution of Sentiments in emotions.csv

2b)

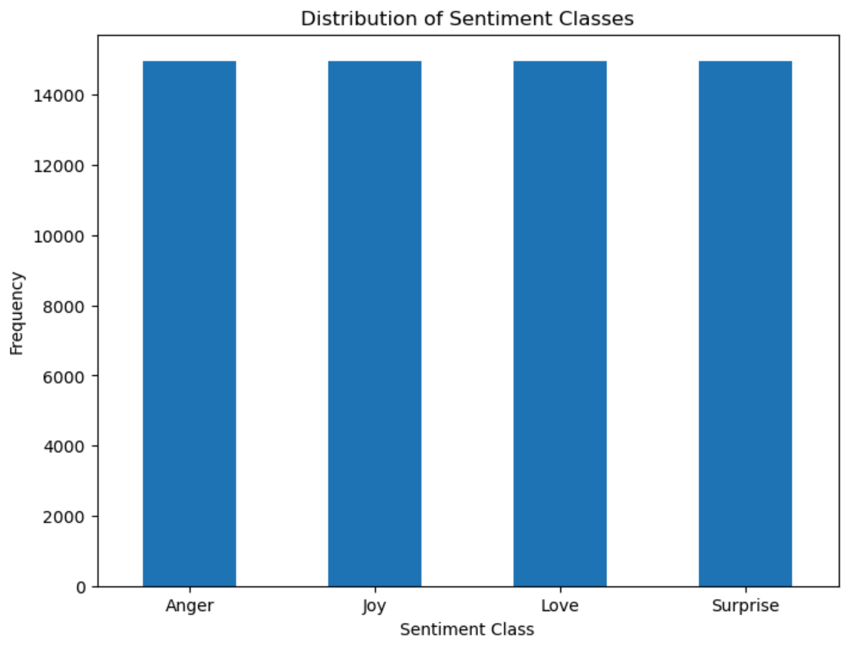


Fig. 2.2 Balanced Dataset After Undersampling

2c)

Total number of tokens before preprocessing data = 1188318

Total number of tokens after preprocessing data = 575823

2d)

After splitting the dataset, the first three lines of training and test corpus are:

Train:

- *"my parents came to visit me"*
- *"ill state that i am angry at myself for feeling…"*
- *"i feel calm complete and whole after i meditate"*

Test:

- *"i feel blessed each day with what i have been …"*
- *"i feel like a loser because all the guys i have..".*
- *"i had a funny feeling when i accepted them"*

2e)

The accuracy of the SGD Classifier is:

Training Accuracy: 0.967

Test Accuracy: 0.953

2f)

- **Generated Documents**:
    - **2-gram**: for the pain you board for a bit off guard
      Classification: Anger
    - **5-gram**: feel like ive gained five pounds this week but was surprised to see that it wasnt just the things that i am interested in the historical role freemasonry had in fathering modern magical orders and modern druid orders but i am also drawn to freemasonrys philosophy of life and self
      Classification: Surprise

- Classification of 1000 documents:

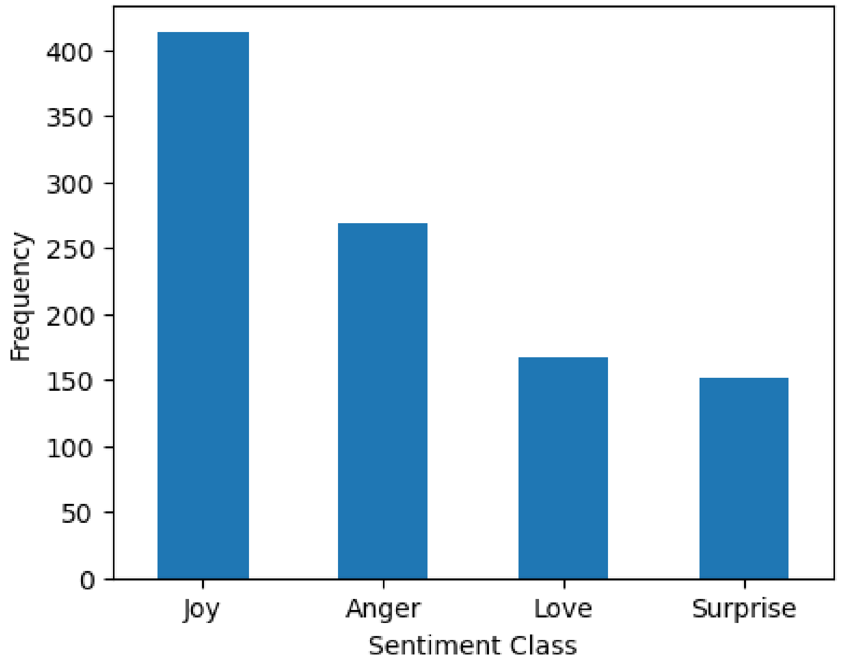


Fig 2.3 2-gram Generated Document Classifications

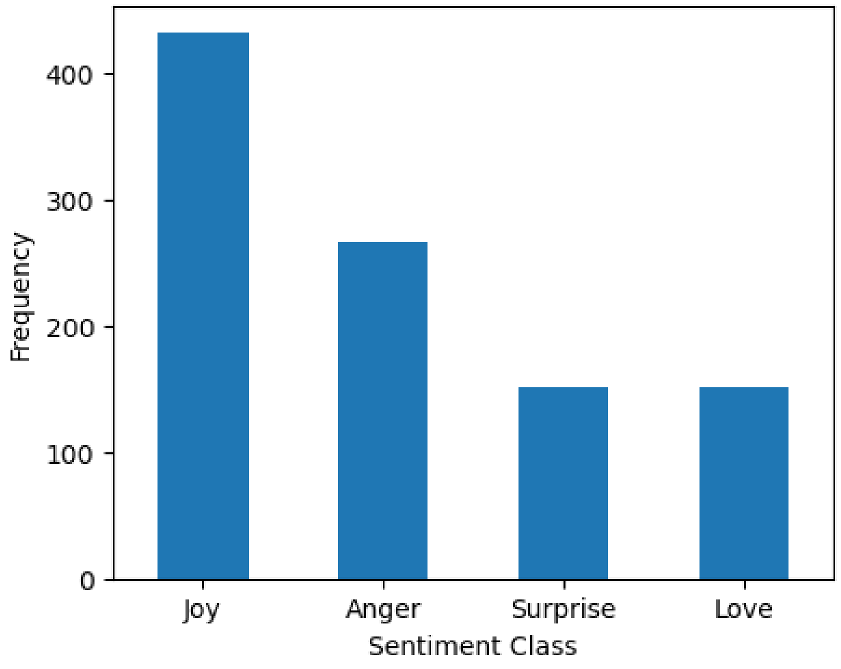


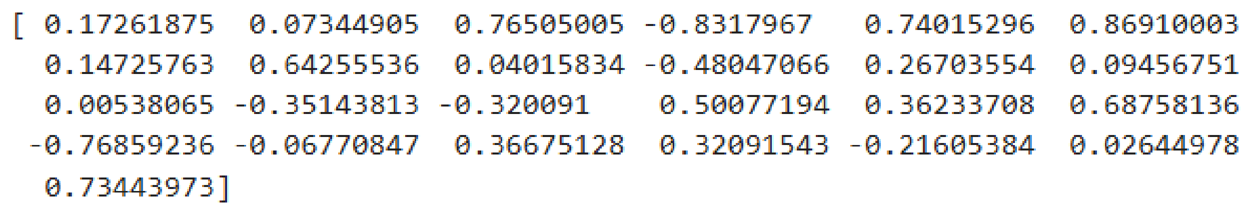
Fig 2.4 5-gram Generated Document Classifications

2g)

The distribution in the 5-gram model seems to be slightly more balanced than 2-gram model. This is because with the increase in n, allows model for a better contextual understanding and reduces the dominance of the most frequent class. However, both the models still show the imbalance present in the original dataset, which shows that the n-gram models do not completely eliminate the original bias.

2h)

Word2Vec embedding for 'christmas':



```
[ 0.17261875  0.07344905  0.76505005 -0.8317967   0.74015296  0.86910003
  0.14725763  0.64255536  0.04015834 -0.48047066  0.26703554  0.09456751
  0.00538065 -0.35143813 -0.320091    0.50077194  0.36233708  0.68758136
 -0.76859236 -0.06770847  0.36675128  0.32091543 -0.21605384  0.02644978
  0.73443973]
```

2i)

Top 3 most similar words to 'christmas':

- holiday= 0.871
- decoration= 0.817
- holiday= 0.808

Top 3 most dissimilar words to 'christmas':

- arrogant= -0.282
- harm= -0.266
- socially= -0.243

High cosine similarity between two embeddings means that the words are used in similar linguistic and contextual environments within our corpus. These words often appear near the same neighbouring words and share semantic and thematic meanings.

2j)

Word2vec organises words in a high-dimensional vector space based on their contextual similarity, so words which appear in similar contexts are closer together. Most similar words will lie close to “christmas” while the dissimilar words will have orthogonal vectors, making their cosine similarity extremely low or negative.

2k)

Top 3 Christmas-Related Emotions:

positive=["christmas", "emotion"]

- excitement= 0.82
- thankfulness= 0.815
- gratitude= 0.812

Top 3 Christmas-Unrelated Emotions:

positive=["emotion"], negative=["christmas"]

- reaction= 0.679
- expression= 0.668
- hatred= 0.65

The most_similar method in Word2Vec computes the similarity of words by first generating a new vector based on the positive and negative keyword embeddings. It does this by summing the embeddings of the positive words and subtracting the embeddings of the negative words, effectively creating a directional semantic meaning in the vector space. The method then finds the closest words to this resulting vector using cosine similarity.

3: Process Mining

3a)

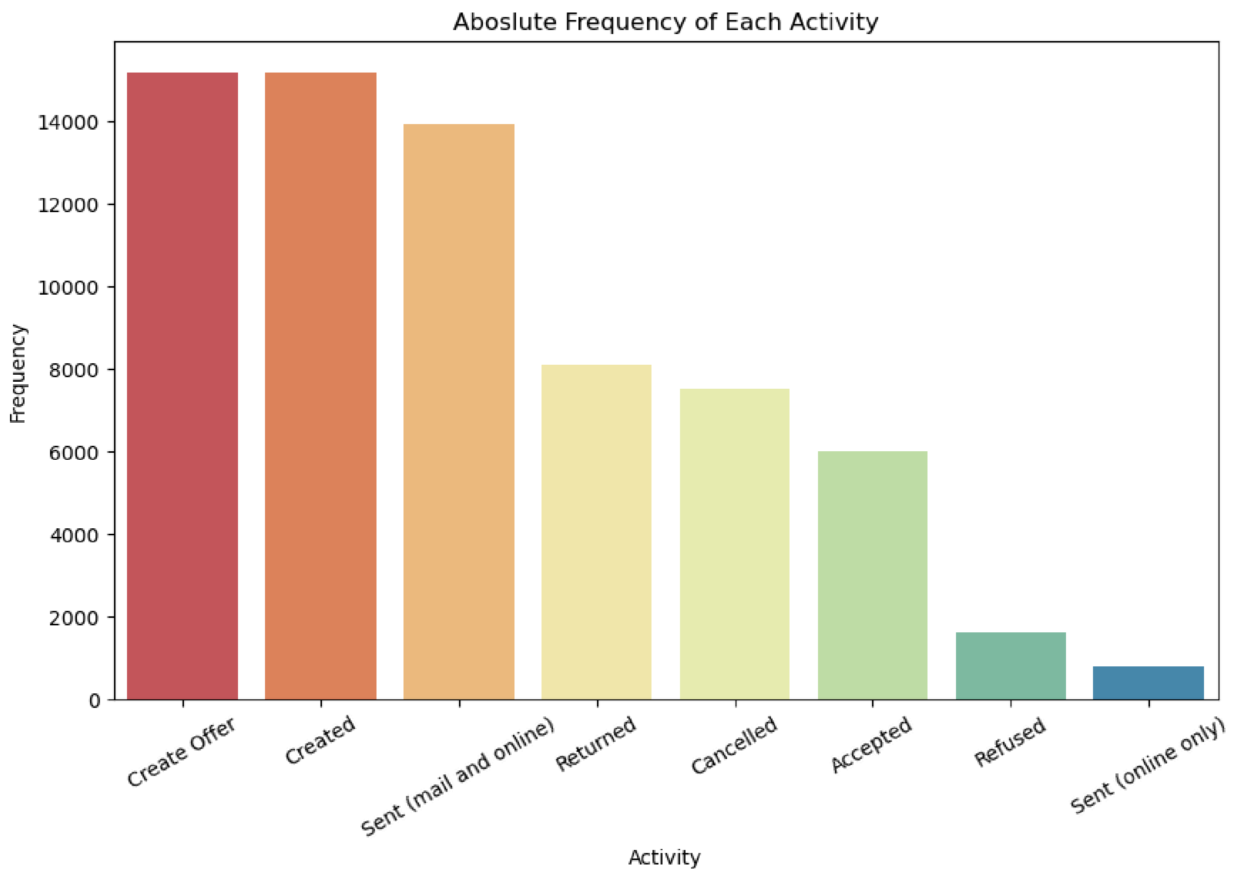


Fig. 3.1 Absolute Frequency of Each Activity

In Fig. 3.1 we can see the eight different activities and their total occurrences. There are 68312 events or entries in the event_log.xes file in total.

3b)

There is a total number of 11021 cases and 15188 offers. Which makes an average of 1.3780963614916977 offers per case.

3c)

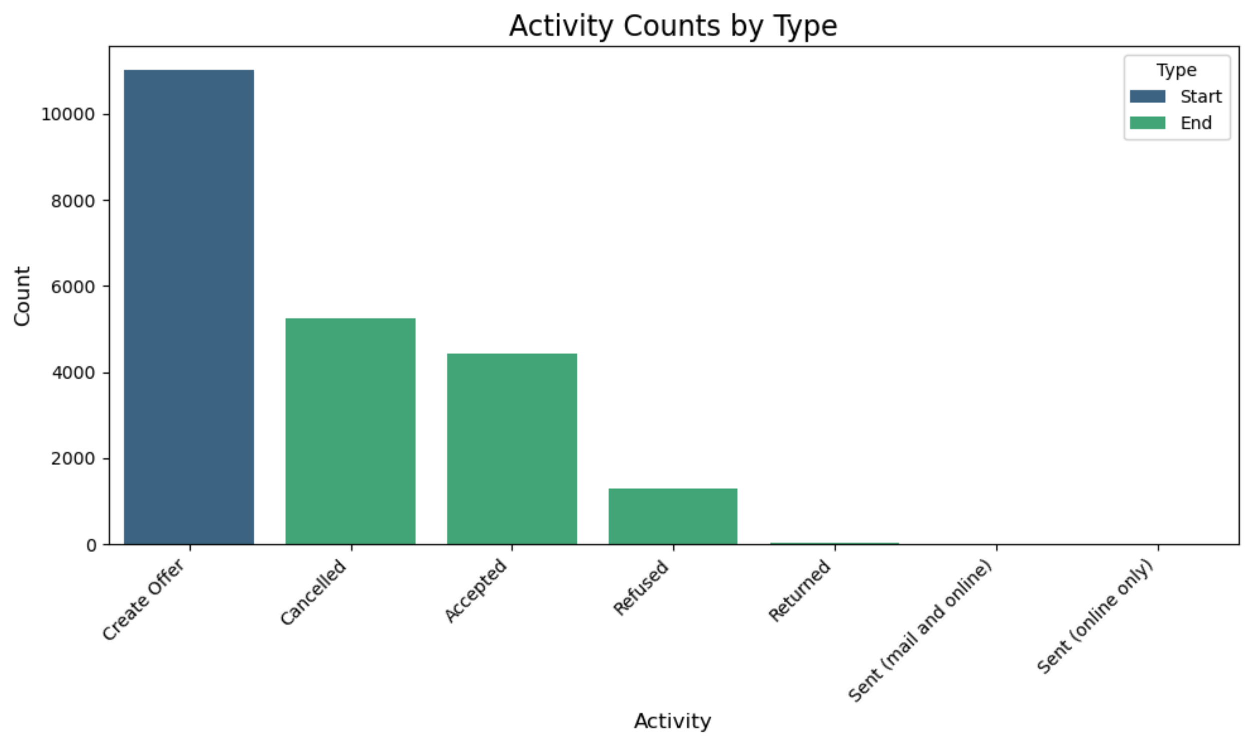


Fig. 3.2 Activity Counts by Type

In Fig 3.2 we can observe, that most processes get cancelled in the beginning or offers get accepted. About 10% get refused and very little get returned or the customer doesn't answer to the offered loan. It might be, that those processes are cut of by taking the data sample in a given timeframe and that they usually would be canceled after the costumer would not respond in time.

3d)

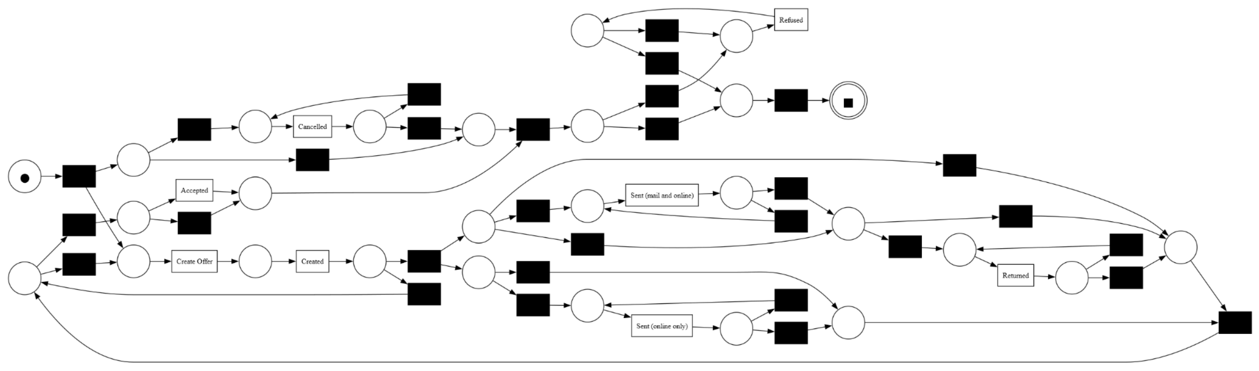


Fig. 3.3 Petri Net Model of the mined Process

Fig. 3.3 shows the petri net discovery by the inductive miner with default settings on the complet log object.

3e)

According to Fig 3.3 the optimal process trace would look like < Create Offer, Created, sent (online only), Accepted>, science the company can only make profit if customers accept their loan offers and it is the shortest trace with a positive outcome. Also sending the offer once online is the process with minimal cost to the company.

3f)

A possible trace, which is not represented is < Create Offer, Created > because “Created” is not an end activity. The Trace is not logical, because creating offers without informing the customer, generates work but no possible revenue.

There are some loops possible, i. E. sending multiple offers before the customer accepted one. In addition there are multiple end activities possible. Because the petri net is correct for all possible end activities the end activities can be bypassed. Our trace creates an offer and bypasses all end activities.

3g)

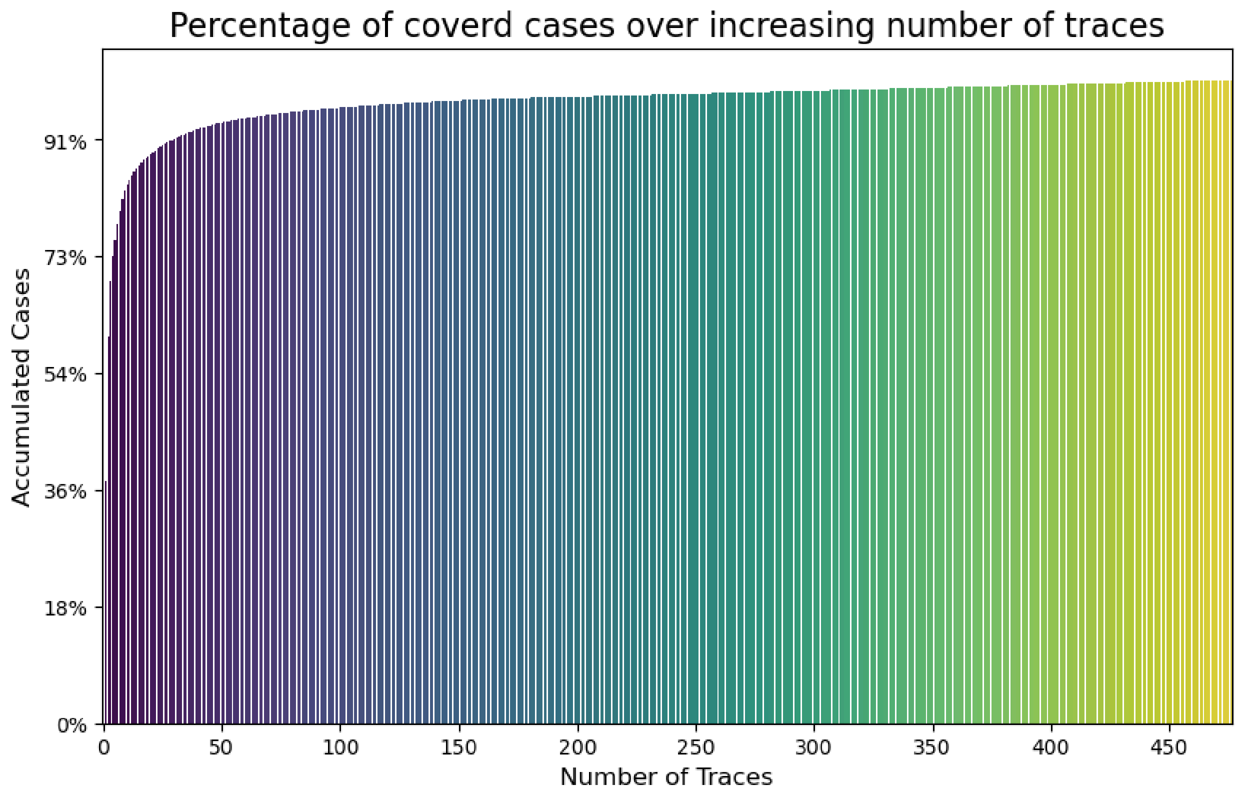


Fig 3.4 Percentage of covered cases over increasing number of traces

3h)

There are a total of 11021 cases in the dataset, 85 % of which is 9367.85 and the top 12 traces cover them. The 2 most common traces cover 6641 cases which is 60.25769 % of the cases.

There are 427 with less than 10 cases.

3i)

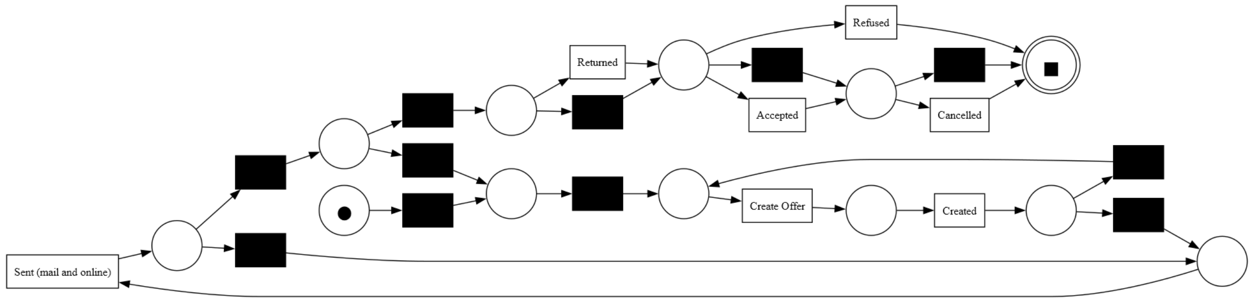


Fig. 3.5 Petri Net of the five most frequent Traces

The petri net of the top 5 traces in fig. 3.5 has a token-based replay fitness of 97,69 % on the complete dataset.

3j)

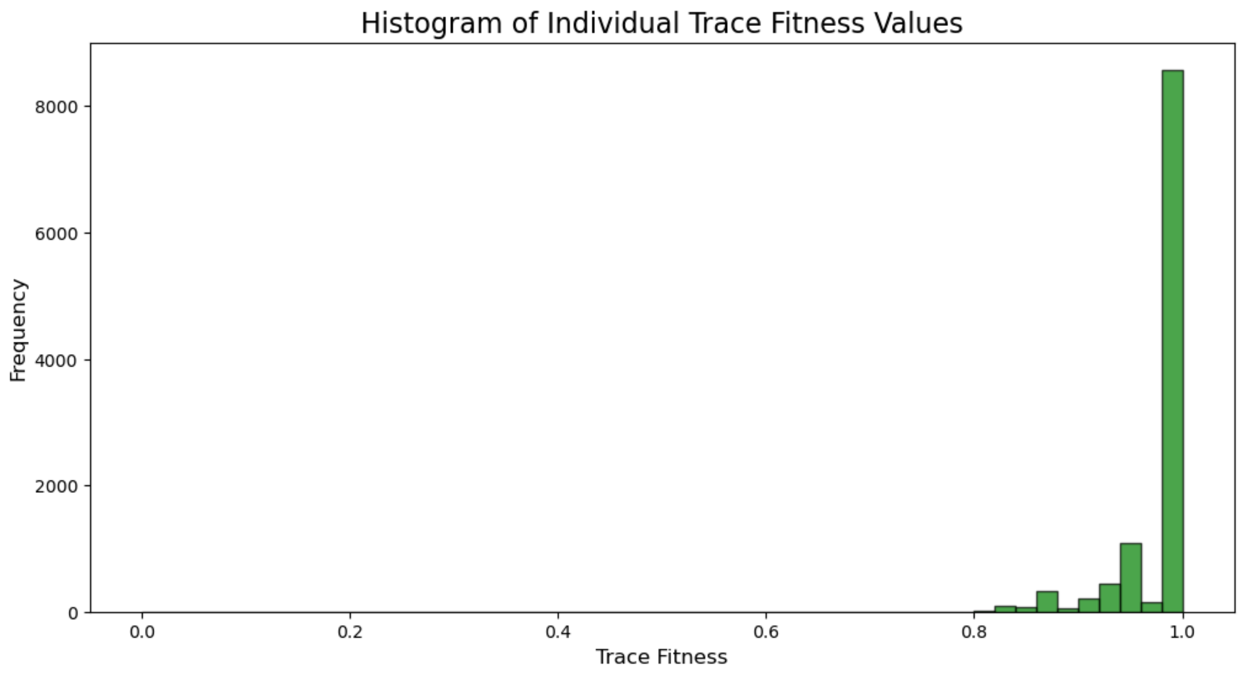


Fig. 3.6 Histogram of Individual Trace Fitness Values

Fig. 3.6 shows that almost all traces have a fitness of 0.98 or more on the petri net from Fig 3.5. Basically all traces have a fitness of more than 0.8.

3k)

The percentage of fitting traces is significantly higher, because the model created by the top traces contains loops i.e. of creating offers and sending them to the customer. In this way all traces containing the variations of those loops are also covered by this petri net.

The Fitness on the full event log is higher than the percentage of fitting traces, because the token-based replay recognizes partially aligned traces.

3l)

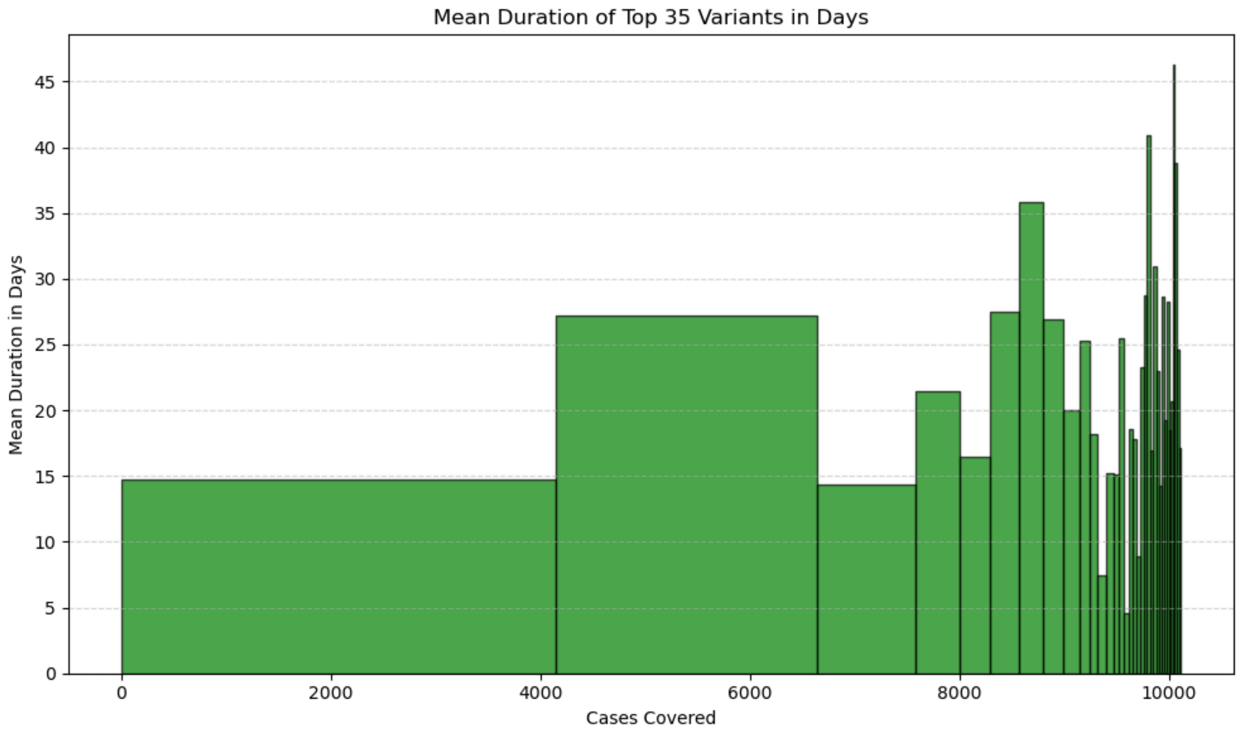


Fig. 3.6 Mean Duration of Top 35 Variants in Days

The mean throughput time of the two most frequent variants is 14.76 and 27.22 days. In Fig. 3.6 we can see that less frequent variants have a larger spread in mean Duration.

3m)

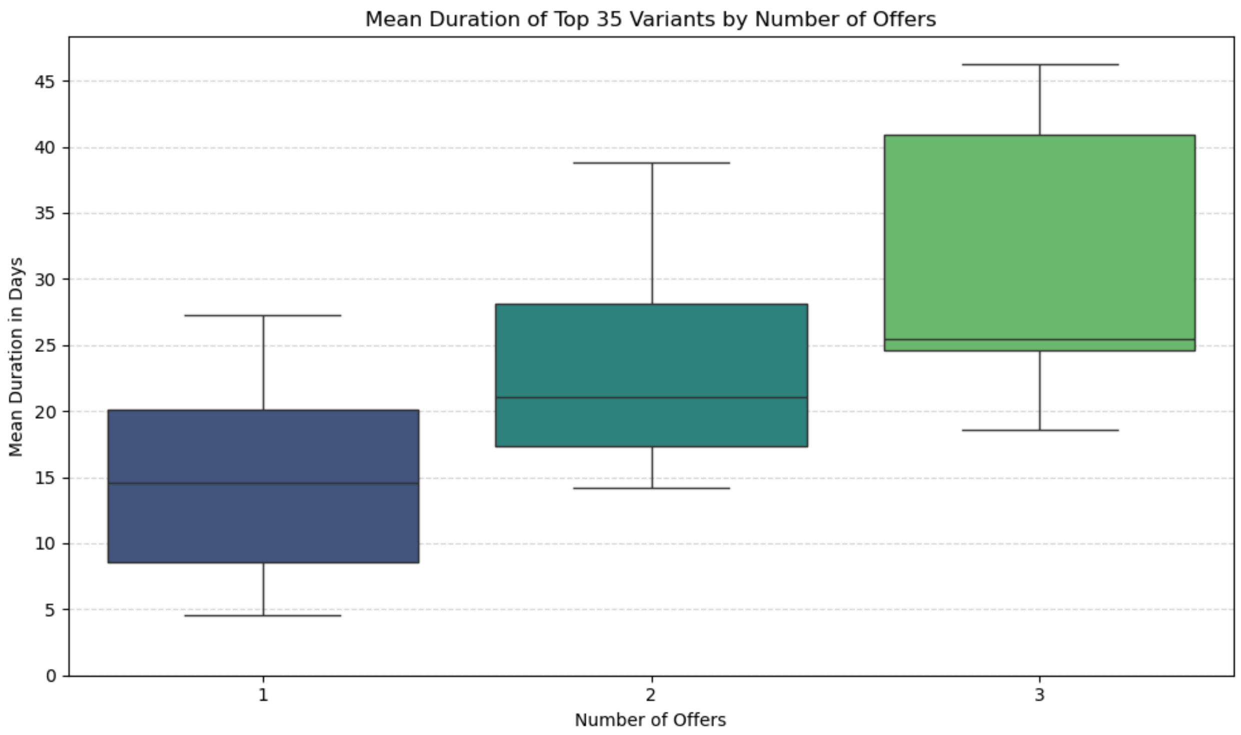


Fig. 3.7 Mean Duration of Top 35 Variants by Number of Offers

In Fig. 3.7 we can see a correlation between the number of offers and an increasing average Duration. Also interesting is that the Median moves closer to the border between the first and second quantile, wich means, that the spread of especially long processing time increases with a higher offer number.

4: Time Series Analysis

4a)

i)

Number of UFO sightings: 80328

Date range: 1906-11-11 00:00:00 to 2014-05-08 18:45:00

ii)

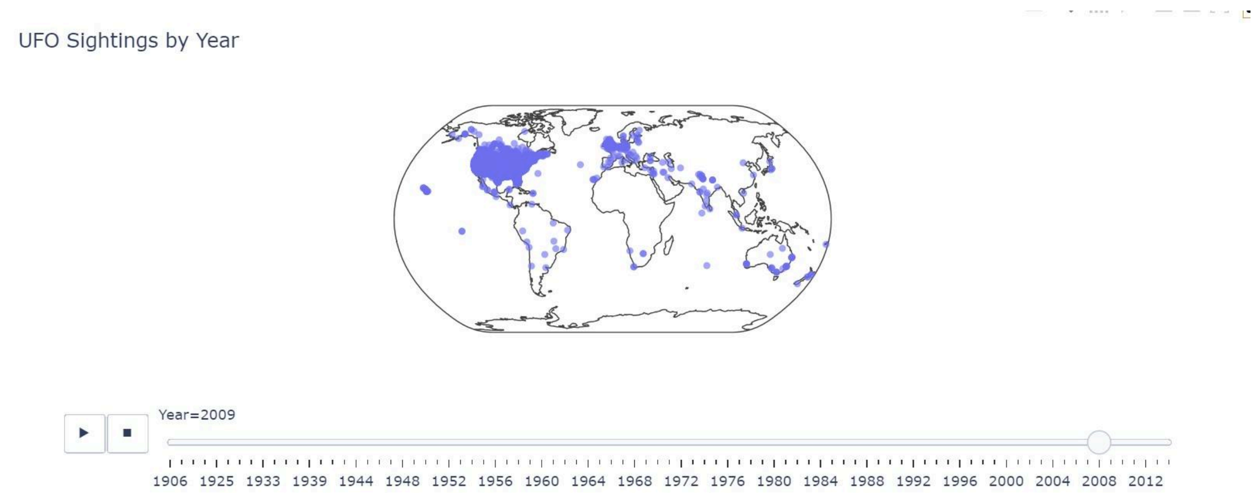


Fig. 4.1: Map of UFO sightings in the world (year 2009 is captured).

The number of UFO sightings has increased over the years. The trend suggests a growing interest or concern about UFOs over time.

iii)

The most prominent trend is that the number of sightings in the United States is significantly higher than in other countries in all years. The number of sightings in the United States has increased over time with a noticeable peak in the 2010s.

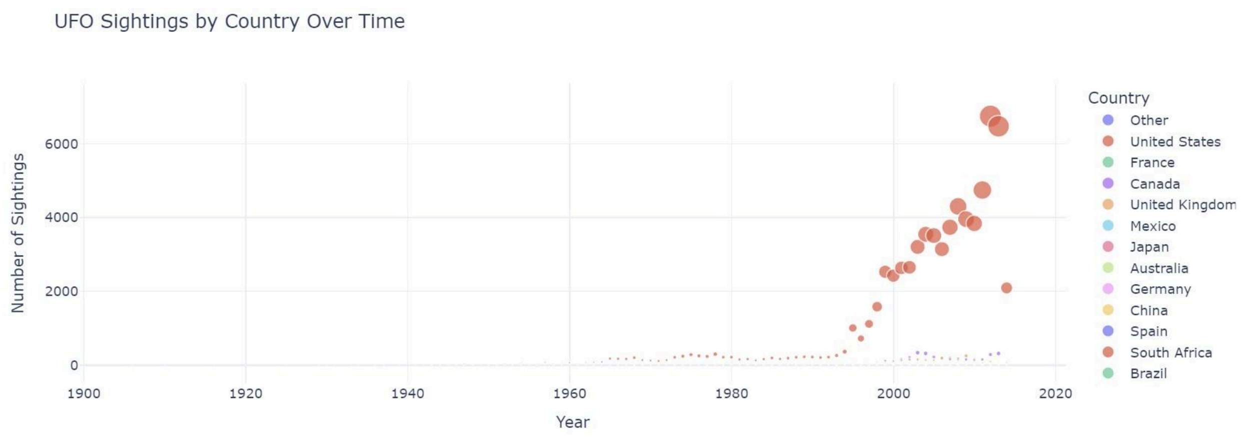
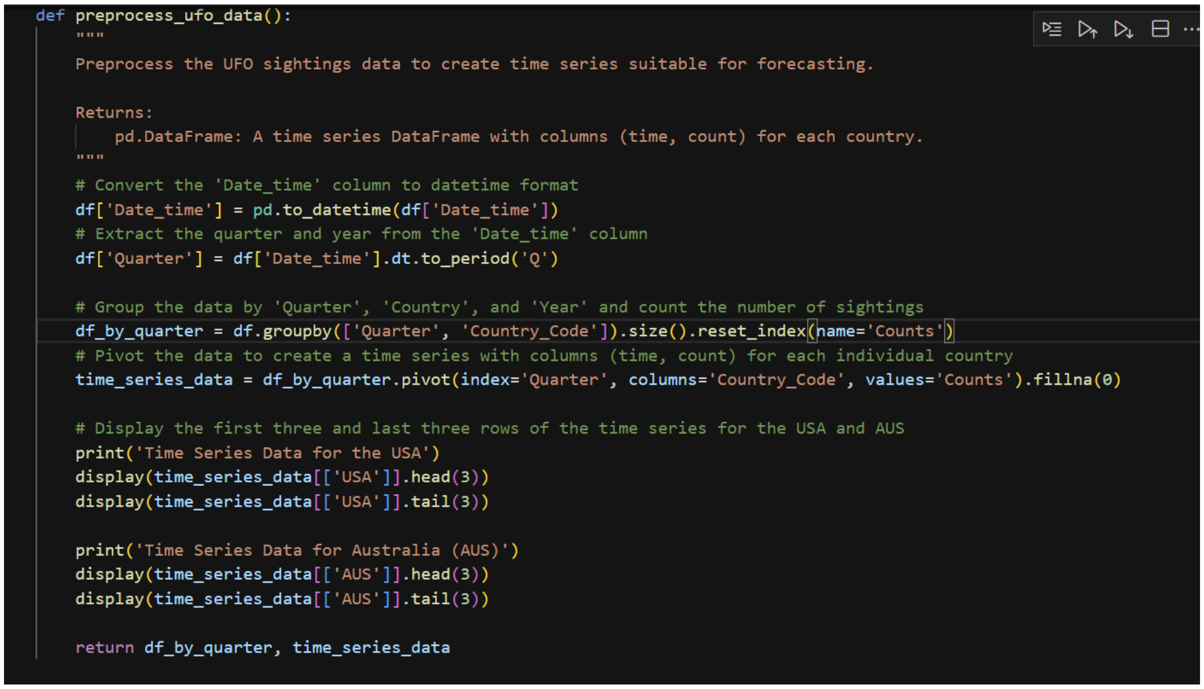


Fig. 4.2: Trend of UFO sightings per country over the years

4b)

i)



```
def preprocess_ufo_data():
    """
    Preprocess the UFO sightings data to create time series suitable for forecasting.

    Returns:
        pd.DataFrame: A time series DataFrame with columns (time, count) for each country.
    """
    # Convert the 'Date_time' column to datetime format
    df['Date_time'] = pd.to_datetime(df['Date_time'])
    # Extract the quarter and year from the 'Date_time' column
    df['Quarter'] = df['Date_time'].dt.to_period('Q')

    # Group the data by 'Quarter', 'Country', and 'Year' and count the number of sightings
    df_by_quarter = df.groupby(['Quarter', 'Country_Code']).size().reset_index(name='Counts')
    # Pivot the data to create a time series with columns (time, count) for each individual country
    time_series_data = df_by_quarter.pivot(index='Quarter', columns='Country_Code', values='Counts').fillna(0)

    # Display the first three and last three rows of the time series for the USA and AUS
    print('Time Series Data for the USA')
    display(time_series_data[['USA']].head(3))
    display(time_series_data[['USA']].tail(3))

    print('Time Series Data for Australia (AUS)')
    display(time_series_data[['AUS']].head(3))
    display(time_series_data[['AUS']].tail(3))

    return df_by_quarter, time_series_data
```

Fig. 4.3: Listing of preprocessing Python functions.

First 3 rows of the time series data for the USA

```
Quarter
1906Q4      0.0
1910Q1      1.0
1910Q2      1.0
```

Last 3 rows of the time series data for the USA

Quarter	
2013Q4	1987.0
2014Q1	1456.0
2014Q2	638.0

First 3 rows of the time series data for Australia (AUS)

Quarter	
1906Q4	0.0
1910Q1	0.0
1910Q2	0.0

Last 3 rows of the time series data for Australia (AUS)

Quarter	
2013Q4	9.0
2014Q1	9.0
2014Q2	5.0

ii)

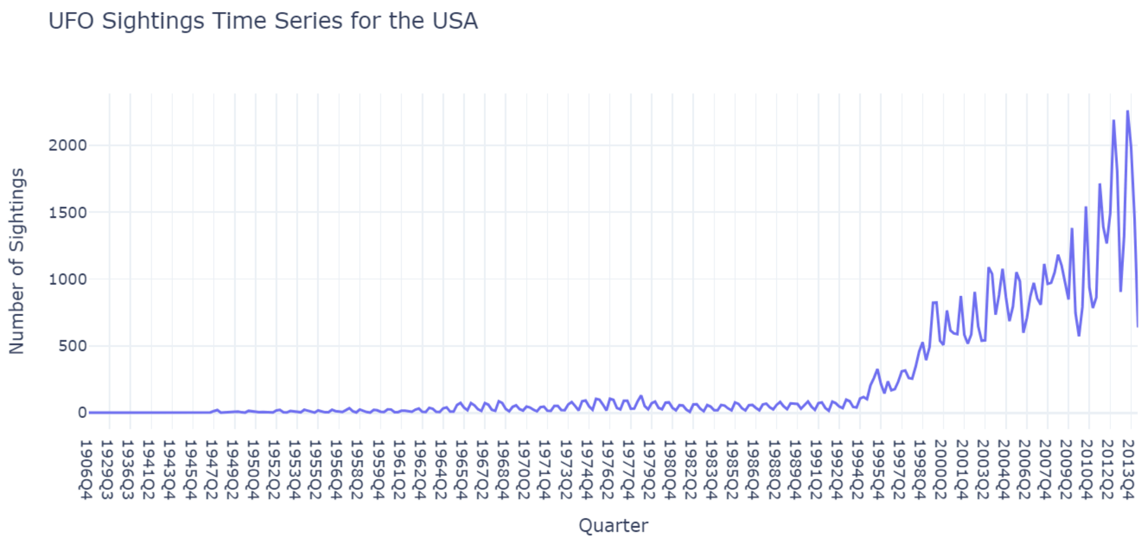


Fig. 4.3: A time series plot for the USA using your preprocessed dataset.

There is a clear upward trend in the number of sightings, with some seasonal variation. The third and fourth quarters of each year seem to have higher sighting counts compared to the first and second quarters. The data also shows some fluctuations, indicating potential anomalies or interesting events.

4c)

i)

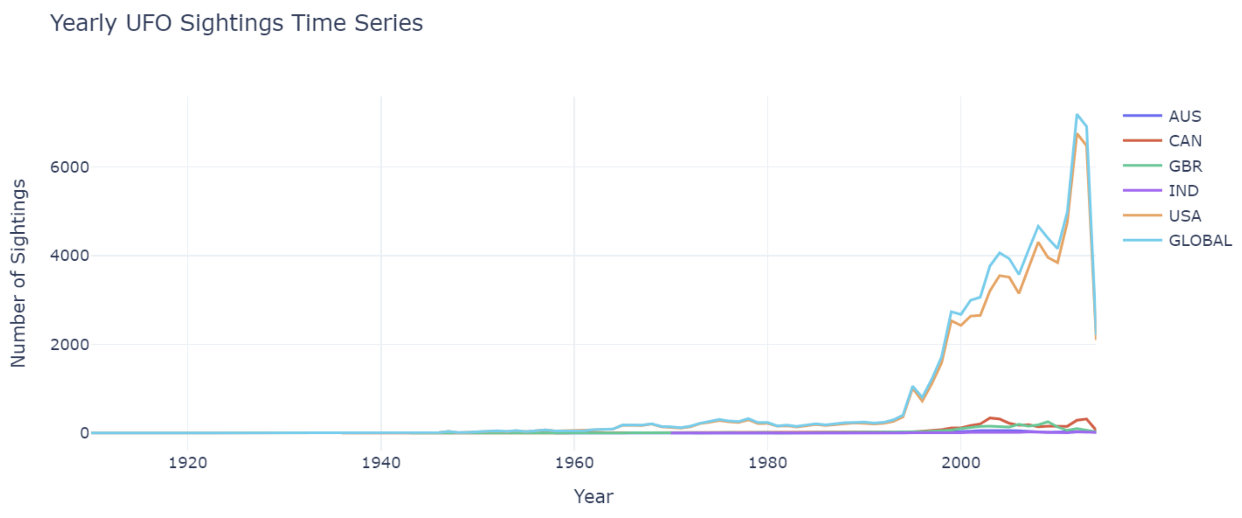


Fig. 4.4: Yearly UFO Sightings Time Series

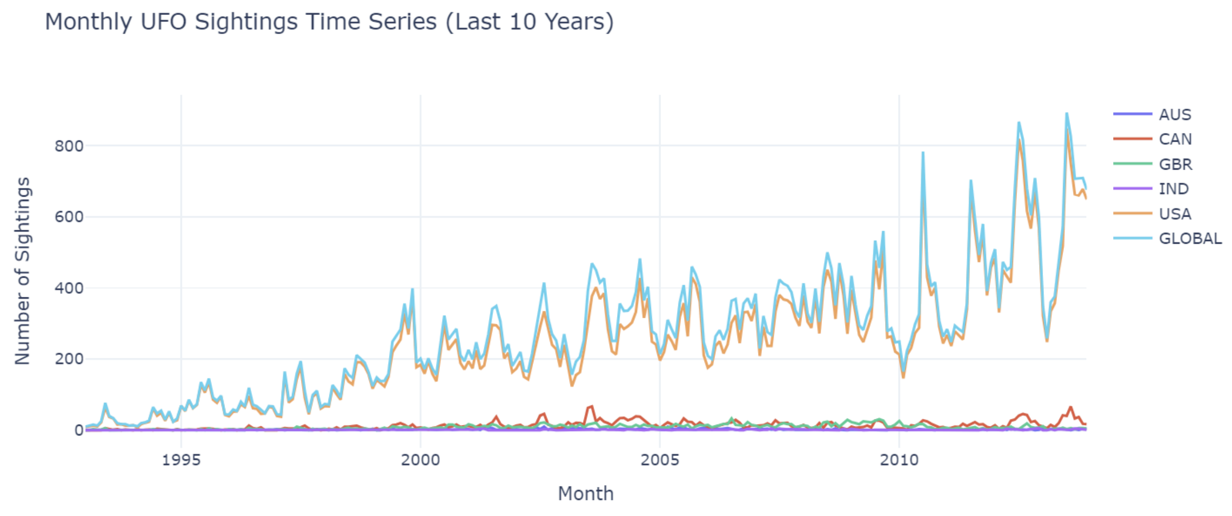


Fig. 4.5: Monthly UFO Sightings Time Series (Last 10 Years)

The major difference between the yearly and monthly time series is the granularity of the data. The yearly time series aggregates the sightings by year, providing a high-level overview of the trends over time. In contrast, the monthly time series captures more detailed variations in the sighting counts, allowing for a closer examination of seasonal patterns or anomalies. The yearly time series plot shows a smoother trend, while the monthly time series plot exhibits more fluctuations and seasonal patterns.

ii)

Yearly Time Series: The yearly time series is better suited for long-term forecasting. Its low granularity makes it ideal for predicting trends over multiple years, though it may miss seasonal patterns. It's suitable for forecasts of the overall trend but lacks the fine details that are necessary for short-term forecasting.

Monthly Time Series: The monthly time series is well-suited for short-term forecasting. The higher frequency allows it to capture seasonal patterns, making it useful for predicting sighting counts within the next few months or recognizing monthly fluctuations. However, its high granularity might require more sophisticated models, especially if seasonality or anomalies are present.

iii)

The sharp drop in the final period (2014) of the yearly time series could be due to incomplete data for that year. In the first part of the question, we saw the last recorded data is in May 2014.

4d)

i)

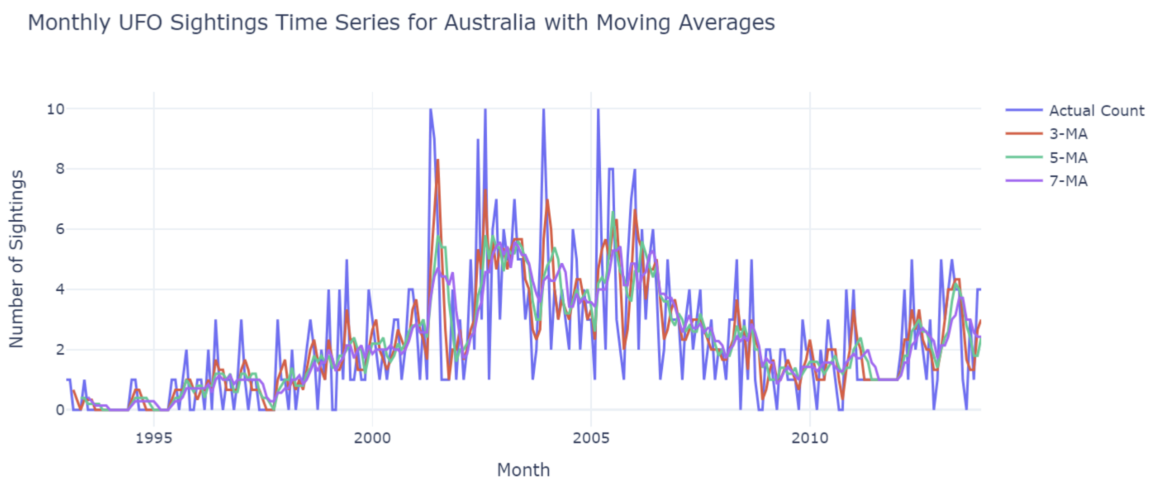


Fig. 4.6: Monthly UFO Sightings Time Series (Last 10 Years)

ii)

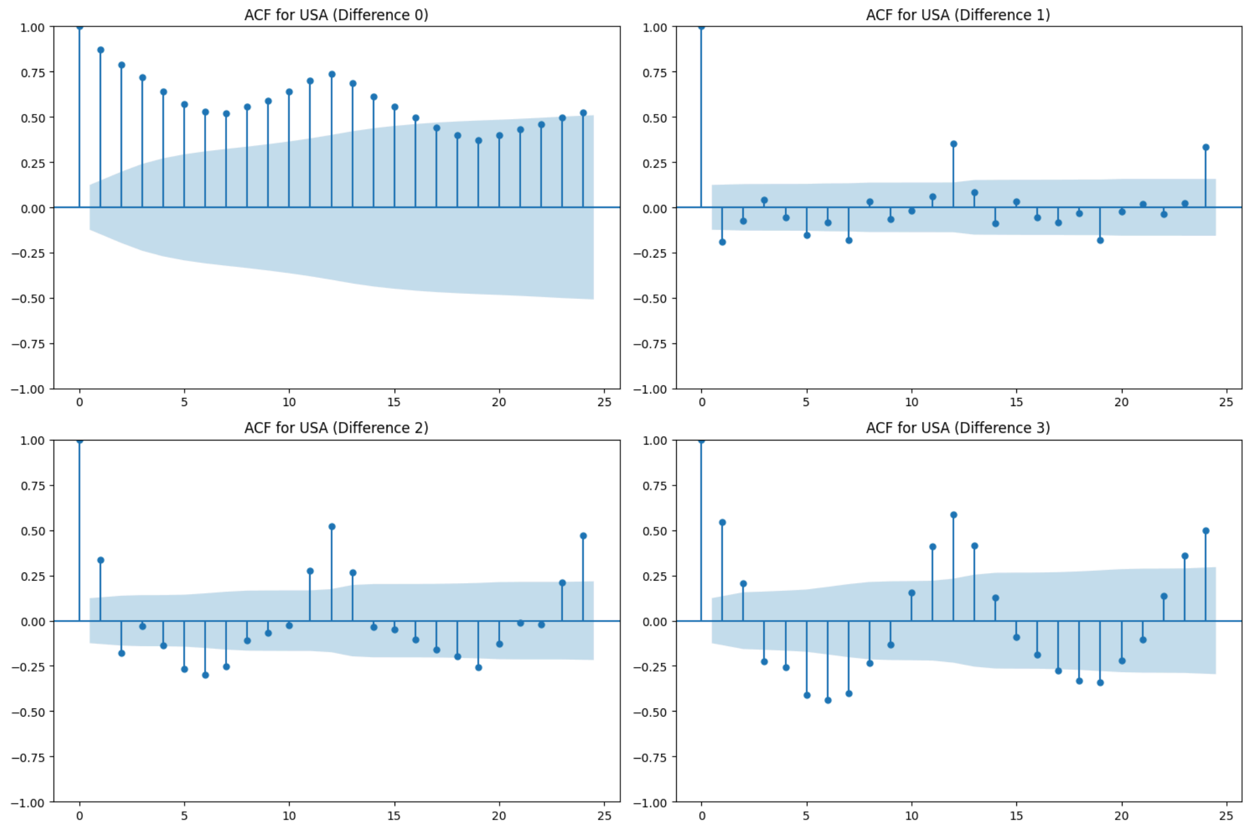


Fig. 4.7: Correlograms for the counts in the USA after differencing

Significant lags are those where the autocorrelation values are outside the shaded region (confidence interval).
For the USA data after differencing:
Difference 0: Significant lags at 12 and 24 months (seasonal pattern)
Difference 1: Significant lags at 1, 12, and 24 months
Difference 2: Significant lags at 1, 12, and 24 months
Difference 3: Significant lags at 1, 12, and 24 months
The seasonal pattern at 12 and 24 months is consistent across the differenced series, indicating a yearly cycle in the data. The significant lags at 1 month suggest a potential monthly pattern or short-term autocorrelation in the data. The autocorrelation plots help identify the lags that may be relevant for forecasting or modeling the time series.

iii)

iv)

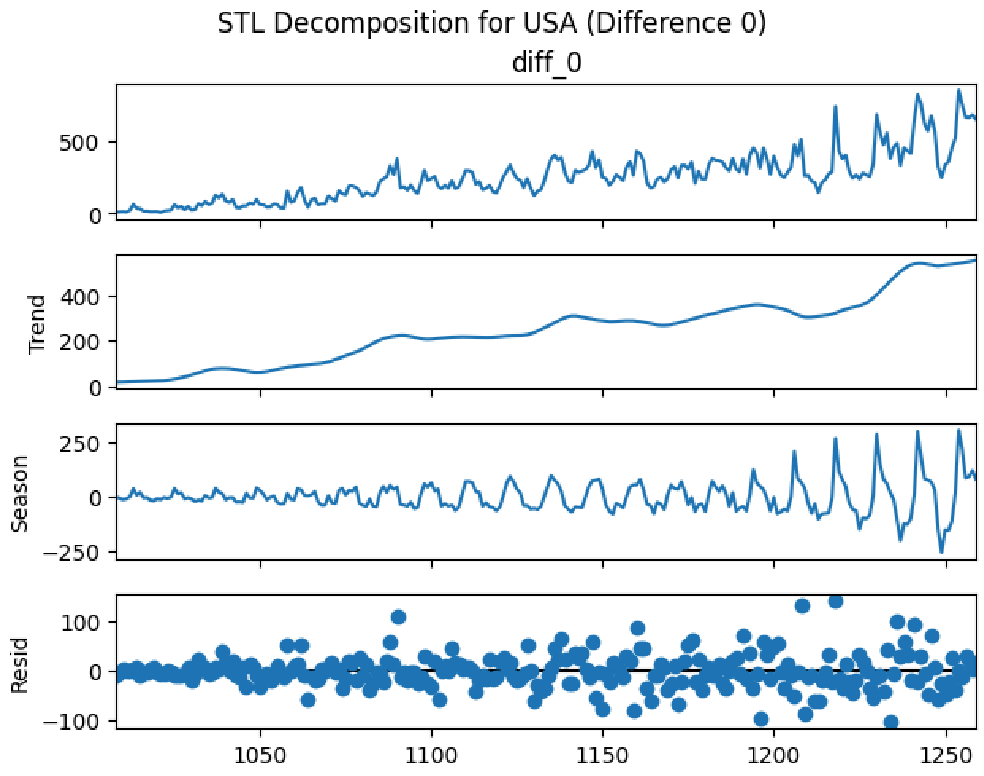


Fig. 4.8

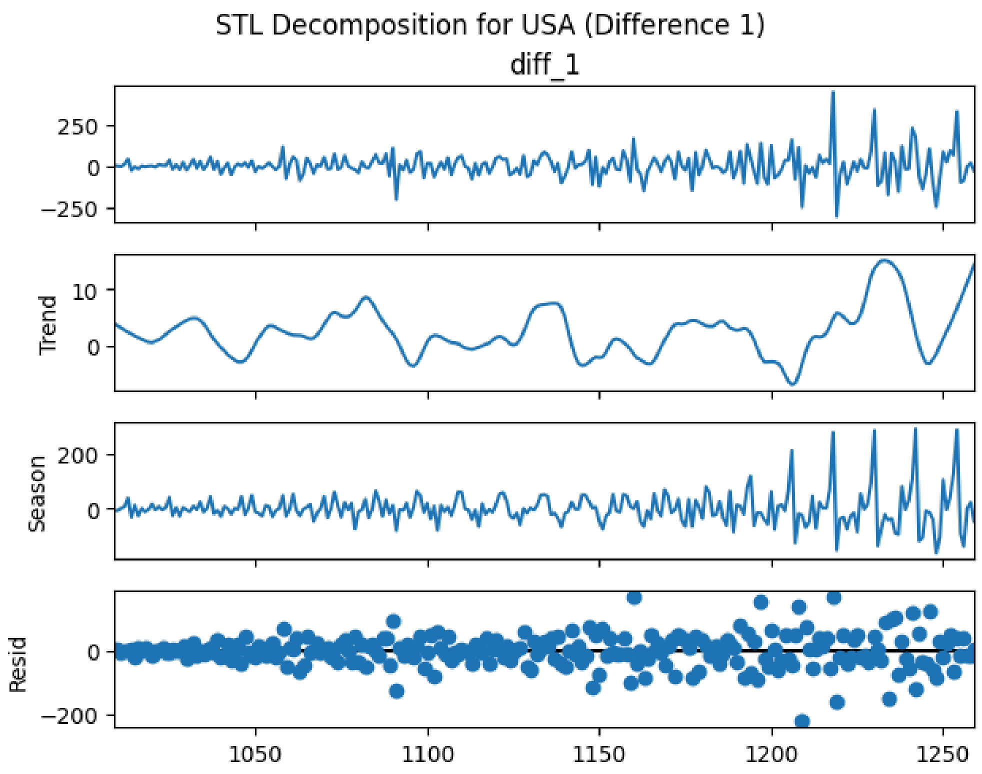


Fig. 4.9

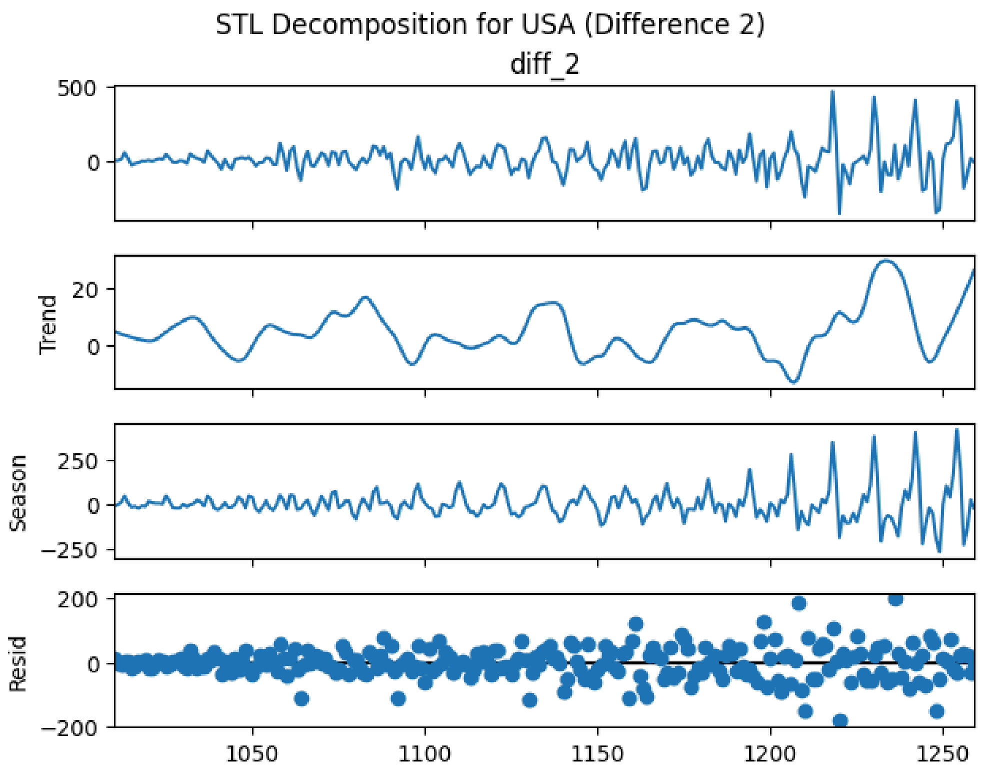


Fig. 4.10

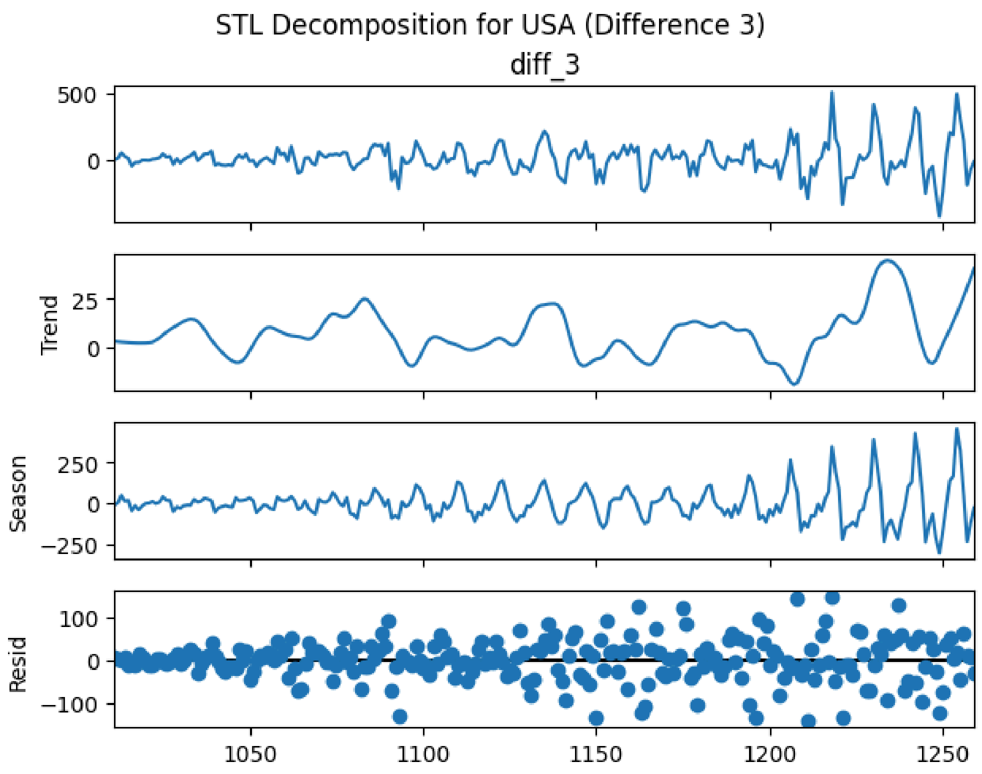


Fig. 4.11

The STL decomposition separates the time series into three components: trend, seasonality, and residual. The trend shows the long-term movement in the data, while the seasonal component captures periodic patterns, and the residual represents the remaining variation after removing both the trend and seasonal effects. To assess stationarity, the differenced series are examined through the residuals from the STL decomposition. If the residuals are stationary, the time series is considered (almost-)stationary after differencing.

The Augmented Dickey-Fuller (ADF) test is used to confirm stationarity. A p-value below 0.05 indicates that the residuals are stationary. Based on this, the first and second differenced series show stationary residuals, making them suitable for further modeling as stationary time series.

4e)

i)

ii)

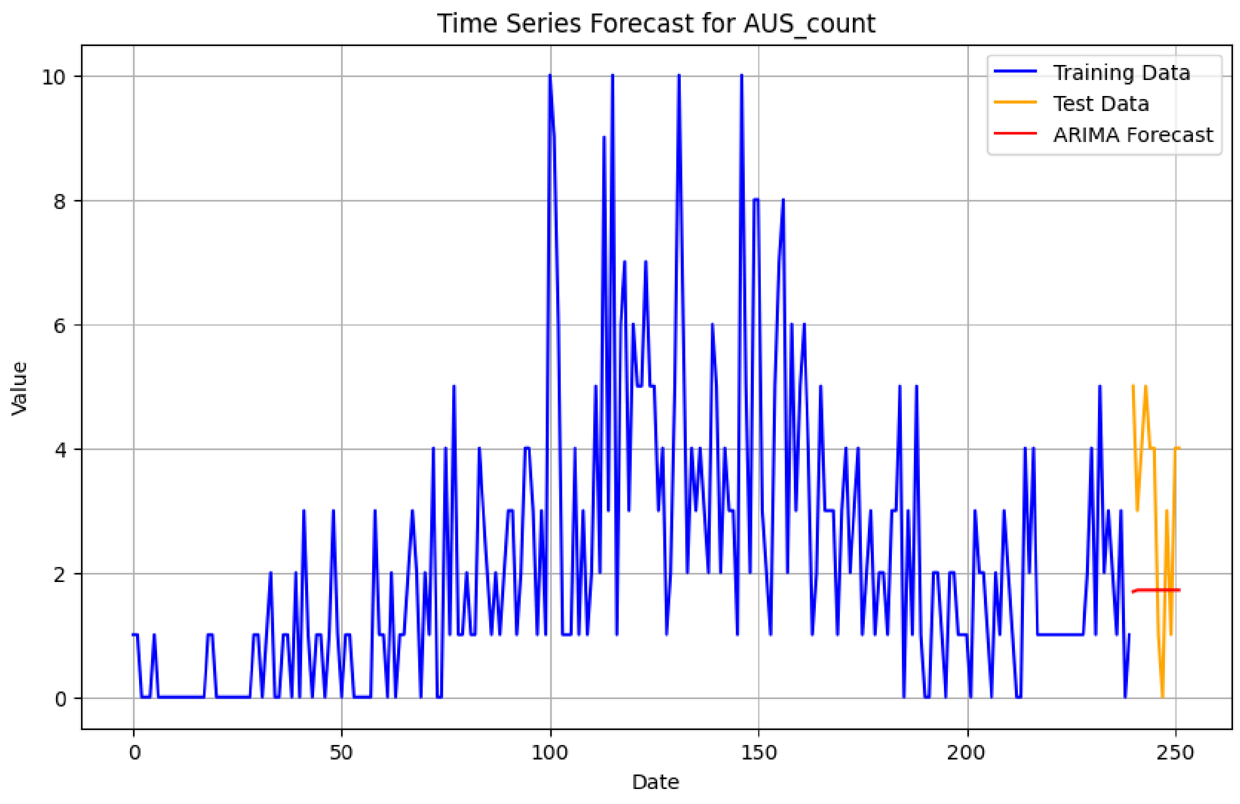


Fig. 4.12: Time series forecast for AUS

iii)

The ARIMA model results vary across countries, with some countries exhibiting better forecasting accuracy than others based on metrics like RMSE, MAE, and MAPE. The forecasting challenges arise from data characteristics such as trends, seasonality, irregular patterns, and data quality. Countries with erratic or irregular patterns, significant outliers, or missing data points pose greater challenges for accurate forecasting. Additionally, unique trends or anomalies in the data further complicate prediction accuracy. Overall, the complexity of the data plays a significant role in the model's performance, affecting its ability to reliably capture patterns and make forecasts.

5: Distributed Data Processing

5a)

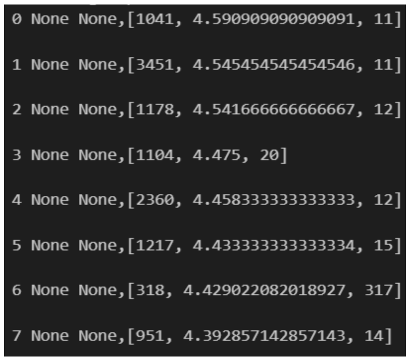
In the following we provide the Mathematical notation for our mapreduce algorithm. In fig. 5.1 and 5.2 you can find the results unedited and as a table for better readability.

Step 1

- ***map*1**: (N_0 x U x M x R x T x R_N) → M × R, N_0
- Input: (N_0 x U x M x R x T x R_N)
- Split rating entry into (movie_id, (rating, 1))
- Return [(movie_id ,(rating, 1))*]
- ***reduce*1**: (movie_id , (rating, 1))* → (movie_id, (sum_ratings ∈ R_N, count_ratings ∈ N_0))
- Input: (movie_id , (rating, 1))*
- Return [(movie_id, (sum_ratings ∈ R_N, count_ratings ∈ N_0))*]

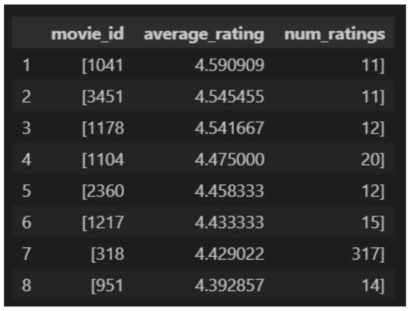
Step 2

- ***map*2**: [(movie_id, (sum_ratings ∈ R_N, count_ratings ∈ N_0))*]
- Input: movie_id, sum_ratings, count_ratings
- average_rating ∈ R_N = sum_ratings / count_ratings
- Return [(movie_id,(average_rating, count_ratings))*]
- ***reduce*2**: (movie_id , (average_rating, count_ratings))* → (movie_id, (sum_ratings ≥ 4.0, count_ratings ≥ 10))
- Input: (movieId, sum_ratings, count_ratings)
- Return [(None,(movieId, avg_rating, count_ratings))*]



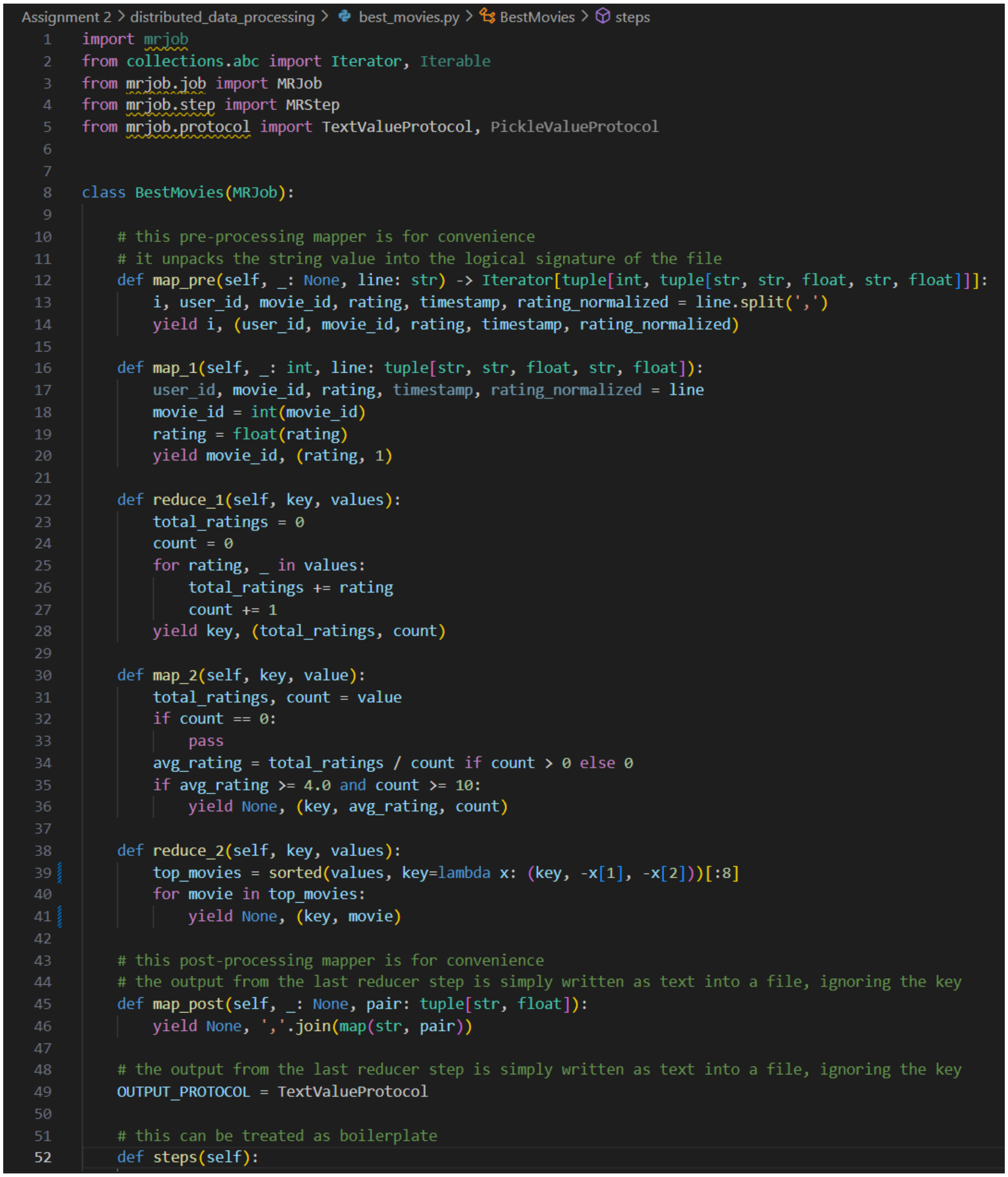
```
0 None None,[1041, 4.590909090909091, 11]

1 None None,[3451, 4.545454545454546, 11]

2 None None,[1178, 4.541666666666667, 12]

3 None None,[1104, 4.475, 20]

4 None None,[2360, 4.458333333333333, 12]

5 None None,[1217, 4.433333333333334, 15]

6 None None,[318, 4.429022082018927, 317]

7 None None,[951, 4.392857142857143, 14]
```

Fig. 5.1 List of top movies



	movie_id	average_rating	num_ratings
1	[1041	4.590909	11]
2	[3451	4.545455	11]
3	[1178	4.541667	12]
4	[1104	4.475000	20]
5	[2360	4.458333	12]
6	[1217	4.433333	15]
7	[318	4.429022	317]
8	[951	4.392857	14]

Figure 5.2 output as Table



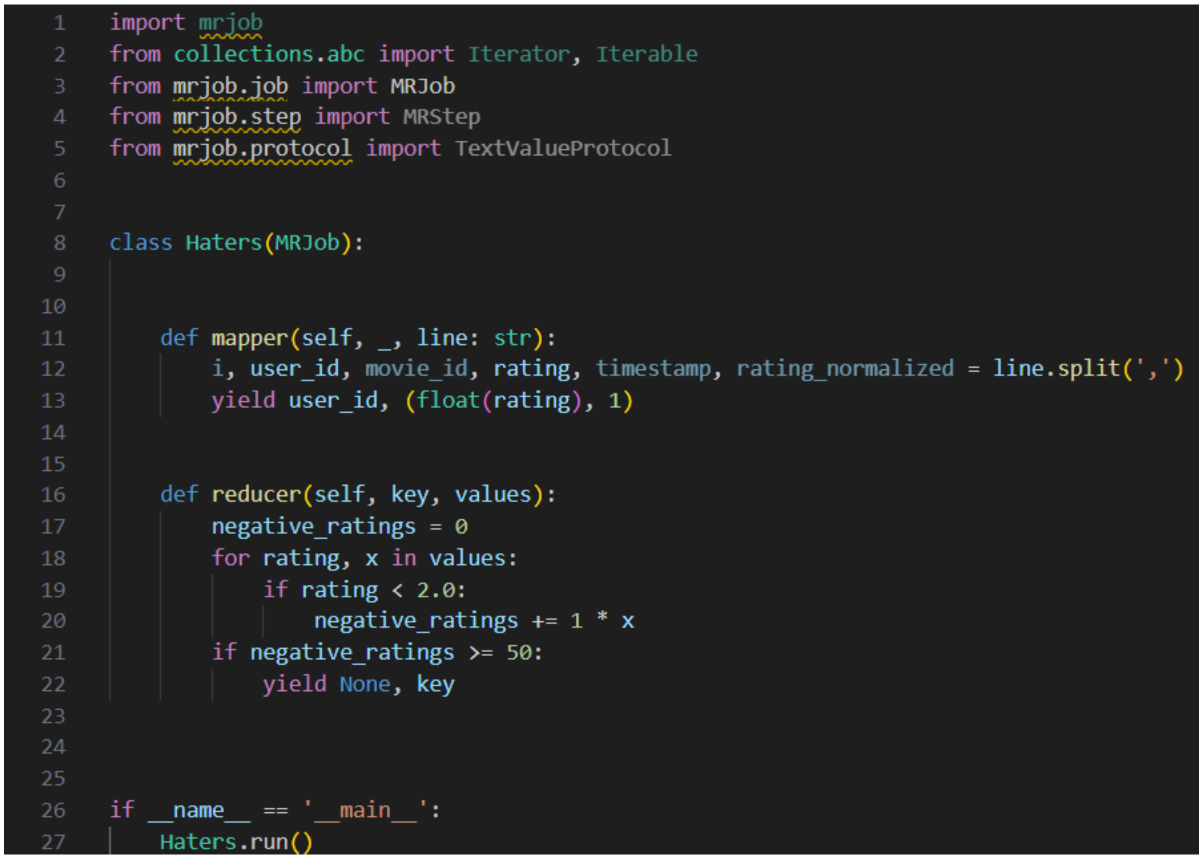
Assignment 2 > distributed_data_processing > best_movies.py > BestMovies > steps

```python
import mrjob
from collections.abc import Iterator, Iterable
from mrjob.job import MRJob
from mrjob.step import MRStep
from mrjob.protocol import TextValueProtocol, PickleValueProtocol


class BestMovies(MRJob):

    # this pre-processing mapper is for convenience
    # it unpacks the string value into the logical signature of the file
    def map_pre(self, _: None, line: str) -> Iterator[tuple[int, tuple[str, str, float, str, float]]]:
        i, user_id, movie_id, rating, timestamp, rating_normalized = line.split(',')
        yield i, (user_id, movie_id, rating, timestamp, rating_normalized)

    def map_1(self, _: int, line: tuple[str, str, float, str, float]):
        user_id, movie_id, rating, timestamp, rating_normalized = line
        movie_id = int(movie_id)
        rating = float(rating)
        yield movie_id, (rating, 1)

    def reduce_1(self, key, values):
        total_ratings = 0
        count = 0
        for rating, _ in values:
            total_ratings += rating
            count += 1
        yield key, (total_ratings, count)

    def map_2(self, key, value):
        total_ratings, count = value
        if count == 0:
            pass
        avg_rating = total_ratings / count if count > 0 else 0
        if avg_rating >= 4.0 and count >= 10:
            yield None, (key, avg_rating, count)

    def reduce_2(self, key, values):
        top_movies = sorted(values, key=lambda x: (key, -x[1], -x[2]))[:8]
        for movie in top_movies:
            yield None, (key, movie)

    # this post-processing mapper is for convenience
    # the output from the last reducer step is simply written as text into a file, ignoring the key
    def map_post(self, _: None, pair: tuple[str, float]):
        yield None, ','.join(map(str, pair))

    # the output from the last reducer step is simply written as text into a file, ignoring the key
    OUTPUT_PROTOCOL = TextValueProtocol

    # this can be treated as boilerplate
    def steps(self):
```

Figure 5.3 Code example of the map reduce algorithm for top 8 movies

In Figure 5.3 you can find the our code we used for our map reduce algorithms.

5b)



```
 1  import mrjob
 2  from collections.abc import Iterator, Iterable
 3  from mrjob.job import MRJob
 4  from mrjob.step import MRStep
 5  from mrjob.protocol import TextValueProtocol
 6
 7
 8  class Haters(MRJob):
 9
10
11      def mapper(self, _, line: str):
12          i, user_id, movie_id, rating, timestamp, rating_normalized = line.split(',')
13          yield user_id, (float(rating), 1)
14
15
16      def reducer(self, key, values):
17          negative_ratings = 0
18          for rating, x in values:
19              if rating < 2.0:
20                  negative_ratings += 1 * x
21          if negative_ratings >= 50:
22              yield None, key
23
24
25
26  if __name__ == '__main__':
27      Haters.run()
```

Fig. 5.4 Code example for map reduce algorithm to find haters

In figure 5.4 we can find the code for the mapper and reducer. By keeping the value for x in line 18 and adding x to negative_ratings in line 20 the code is optimized for combiners. The following table is the output of the code above:

```
0 None 111
1 None 139
2 None 153
3 None 160
4 None 182
5 None 19
6 None 219
7 None 274
8 None 287
9 None 294
```

5c)

The algorithm in Fig 5.4 would work best if the data is spread evenly over a large number of nodes, so that the network infrastructure during the shuffling process would be the biggest bottleneck. In that case a combiner would reduce the network traffic by summarizing all values with the same key, so that the keys on each nodes are unique and their values would represent the sums or averages or counts of all previous values with the same key, depending on the reduce function.